

IN THE CLAIMS:

1 1-66. (CANCELLED)

1 67. (PREVIOUSLY PRESENTED) A processor, comprising:

2 a first execution unit having a first and second input register coupled to first and
3 second inputs to a first arithmetic logic unit (ALU), the first and second input registers of
4 the first execution unit to store source operands, the first ALU capable of addressing a
5 memory to retrieve a source operand;

6 a second execution unit having a first and second input register, the second register
7 coupled to a second input to a second ALU, the first and second input registers of the
8 second execution unit to store source operands, the second ALU not capable of address-
9 ing the memory; and

10 a multiplexer (MUX) having i) a first input coupled to the first input of the first
11 ALU, ii) a second input coupled to the first input register of the second ALU, and iii) an
12 output directly providing a first input to the second ALU, the MUX permitting both the
13 first and second ALU to share the source operand retrieved by the first ALU from the
14 memory.

1 68. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:

2 an instruction set defining a register decode value that specifies source operand
3 bypassing, such that the MUX, in response to the register decode value that specifies
4 source operand bypassing, selects the first input of the MUX coupled to the first input of
5 the first ALU as the output of the MUX, the output of the MUX providing the first input
6 to the second ALU.

- 1 69. (PREVIOUSLY PRESENTED) The processor of claim 68, wherein the source op-
2 erand bypassing value allows the second execution unit to receive data stored at an effec-
3 tive memory address specified by a displacement operand in the previous instruction exe-
4 cuted by the first execution unit.

- 1 70. (CURRENTLY AMENDED) The processor of claim 67, further comprising:
2 a local bus for communicating with a the memory;
3 a register file for storing intermediate operands; and
4 an instruction decode stage for coupling the register file to the first and second in-
5 put registers of the first and second ALUs to provide intermediate operands as the source
6 operands, and for coupling a memory bus to the first input register of the of the first ALU
7 to provide source operands from the memory.

- 1 71. (PREVIOUSLY PRESENTED) The processor of claim 70, wherein the first input
2 register of the first ALU provides source operands from the memory to both the first in-
3 put to the first ALU and to the first input of the MUX, thereby permitting the first input
4 to the second ALU to share the source operands from the memory directly from the first
5 input register of the first ALU.

- 1 72. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:
2 a pipeline of the processor, the pipeline having a plurality of stages including in-
3 struction decode, writeback, and execution stages, the execution stage having the first and
4 second execution units.

- 1 73. (PREVIOUSLY PRESENTED) The processor of claim 72, further comprising:
 - 2 an instruction set defining a register decode value that defines result bypassing
 - 3 that allows bypassing of a result from a previous instruction executing in pipeline stages
 - 4 of the processor by directly addressing a result register of the first and second execution
 - 5 units.

- 1 74. (PREVIOUSLY PRESENTED) The processor of claim 73, wherein the register de-
2 code value comprises:
 - 3 one of a result bypass (RRB) operand and an inter-unit result bypass (RIRB) op-
4 erand, each of which explicitly controls data flow within the pipeline of the processor.

- 1 75. (PREVIOUSLY PRESENTED) The processor of claim 74, wherein the RRB oper-
2 and denotes the first execution unit and the RIRB operand denotes the second execution
3 unit.

- 1 76. (PREVIOUSLY PRESENTED) The processor of claim 74, wherein the RRB oper-
2 and explicitly infers feedback of the data delivered from the first execution unit to an in-
3 put register of the first execution unit over a feedback path.

- 1 77. (PREVIOUSLY PRESENTED) The processor of claim 76, wherein the writeback
2 stage comprises an interstage register and wherein the RRB operand enables bypassing
3 write-back of the data processed by the first and second execution units to one of the reg-
4 ister file or the interstage register of the writeback stage.

- 1 78. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:

2 an instruction set defining a register decode value that defines source operand by-
3 passing that allows source operand data to be shared among the first and second execu-
4 tion units by directly addressing a source register of the first execution unit.

1 79. (PREVIOUSLY PRESENTED) The processor of claim 78, wherein the source op-
2 erand bypassing value allows the second execution unit to receive data stored at an effec-
3 tive memory address specified by a displacement operand in the previous instruction exe-
4 cuted by the first execution unit.

1 80. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:
2 a register file containing a plurality of general-purpose registers for storing inter-
3 mediate result data processed by the first and second execution units.

1 81. (PREVIOUSLY PRESENTED) The processor of claim 67, wherein the first and
2 second execution units are parallel execution units.

1 82. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:
2 a current execution unit as the first execution unit; and
3 an alternate execution unit as the second execution unit.

1 83. (PREVIOUSLY PRESENTED) The processor of claim 67, wherein the first and
2 second ALU share the source operand stored in the first input register of the first ALU
3 substantially simultaneously.

1 84-91. (CANCELLED)

1 92. (PREVIOUSLY PRESENTED) A method for use with a processor, the method
2 comprising:

3 providing a multiplexer (MUX) having a first and second MUX input and a MUX
4 output;

5 coupling the first MUX input to a first input of a first ALU, the first ALU capable
6 of addressing a memory to retrieve a source operand from the memory;

7 coupling a second MUX input to a first input register of a second ALU, the sec-
8 ond ALU not capable of addressing the memory; and

9 directly providing, by the MUX output, a first input to the second ALU, the MUX
10 permitting both the first and second ALU to share the source operand retrieved from the
11 memory by the first ALU.

1 93. (PREVIOUSLY PRESENTED) The method of claim 92, further comprising:

2 defining a register decode value within an instruction set that specifies source op-
3 erand bypassing, such that the MUX, in response to the register decode value that speci-
4 fies source operand bypassing, selects the first MUX input coupled to the first input of
5 the first ALU as the MUX output, the MUX output providing the first input to the second
6 ALU.

1 94. (PREVIOUSLY PRESENTED) The method of claim 93, wherein the source oper-
2 erand bypassing value allows the second execution unit to receive data stored at an effec-
3 tive memory address specified by a displacement operand in the previous instruction exe-
4 cuted by the first execution unit.

1 95. (CURRENTLY AMENDED) The method of claim 92, further comprising:
2 communicating with a the memory;
3 storing intermediate operands in a register file; and
4 identifying an instruction decode stage for coupling the register file to the first
5 and second input registers of the first and second ALUs to provide intermediate operands
6 as source operands, and for coupling a memory bus to the first input register of the of the
7 first ALU to provide source operands from the memory.

1 96. (PREVIOUSLY PRESENTED) The method of claim 95, further comprising:
2 providing, by the first input register of the first ALU, source operands from the
3 memory to both the first input to the first ALU and to the first MUX input, thereby per-
4 mitting the first input to the second ALU to share the source operands from the memory
5 directly from the first input register of the first ALU.

1 97. (PREVIOUSLY PRESENTED) The method of claim 92, further comprising:
2 including a pipeline of the processor, the pipeline having a plurality of stages in-
3 cluding instruction decode, writeback, and execution stages, the execution stage having a
4 first and second execution unit, each having one of the first and second ALUs, respec-
5 tively.

1 98. (PREVIOUSLY PRESENTED) The method of claim 97, further comprising:
2 defining a register decode value that defines result bypassing of a result from a
3 previous instruction executing in pipeline stages of the processor.

1 99. (PREVIOUSLY PRESENTED) The method of claim 98, further comprising:

2 identifying a pipeline stage register for use as a source operand in an instruction
3 containing the register decode value by directly addressing a result register.

1 100. (PREVIOUSLY PRESENTED) The method of claim 99, further comprising:
2 explicitly controlling data flow within the pipeline stages of the processor through
3 the use of a register result bypass (RRB) operand in the register decode value.

1 101. (PREVIOUSLY PRESENTED) The method of claim 100, wherein the step of ex-
2 plicitly controlling comprises:

3 retrieving data from the first execution unit; and
4 returning the data to an input of the first and second execution units as specified
5 by the RRB operand, thereby bypassing write-back of the data to either a register file or
6 memory at the writeback stage.

1 102. (PREVIOUSLY PRESENTED) The method of claim 101, wherein the step of
2 identifying further comprises:

3 explicitly specifying the pipeline stage register to be used as the source operand for
4 the instruction.

1 103. (PREVIOUSLY PRESENTED) The method of claim 102, further comprising:
2 encoding the RRB operand in fewer bits than a regular register operand.

1 104. (PREVIOUSLY PRESENTED) The method of claim 97, further comprising:

2 defining a register decode value that defines source operand bypassing of source
3 operand data.

1 105. (PREVIOUSLY PRESENTED) The method of claim 104, further comprising:
2 identifying a pipeline stage register for use as a source operand in an instruction
3 containing the register decode value by directly addressing a source register.

1 106. (PREVIOUSLY PRESENTED) The method of claim 105, further comprising:
2 sharing source operand data among the first and second execution units of the
3 pipelined processor through the use of a source bypass (RISB) operand in the register de-
4 code value.

1 107. (PREVIOUSLY PRESENTED) The method of claim 106, wherein the step of
2 sharing further comprises:
3 receiving data at the second execution unit, the data stored at a memory address
4 specified by a displacement operand in a previous instruction executed by the first execu-
5 tion unit of the processor.

1 108. (PREVIOUSLY PRESENTED) The method of claim 107, wherein the step of
2 sharing further comprises:
3 realizing two memory references through the use of a single bus operation over a
4 local bus.

1 109. (PREVIOUSLY PRESENTED) The method of claim 108, wherein the step of
2 sharing further comprises:

3 encoding the RISB operand with substantially fewer bits than those needed for a
4 displacement address.

1 110. (PREVIOUSLY PRESENTED) The method of claim 92, further comprising:
2 sharing a source operand stored in a first input register of the first ALU at the first
3 and second ALU substantially simultaneously.

1 111. (CURRENTLY AMENDED) An apparatus, comprising:
2 means for providing a multiplexer (MUX) having a first and second MUX input
3 and a MUX output;

4 means for coupling the first MUX input to a first input for a first ALU, the first
5 ALU capable of addressing a memory to retrieve a source operand from the memory;
6 means for coupling a second MUX input to a first input register for a second
7 ALU, the second ALU not capable of addressing the memory; and

8 means for directly providing, by the MUX output, a first input to the second ALU,
9 the MUX-means for directly providing permitting both the first and second ALU to share
10 the source operand retrieved from the memory by the first ALU.

1 112. (PREVIOUSLY PRESENTED) The apparatus of claim 111, further comprising:
2 means for defining a register decode value within an instruction set that specifies
3 source operand bypassing, such that the MUX, in response to the register decode value
4 that specifies source operand bypassing, selects the first MUX input coupled to the first
5 input of the first ALU as the MUX output, the MUX output providing the first input to
6 the second ALU.

1 113. (PREVIOUSLY PRESENTED) The apparatus of claim 112, wherein the source
2 operand bypassing value allows the second execution unit to receive data stored at an ef-
3 fective memory address specified by a displacement operand in the previous instruction
4 executed by the first execution unit.

1 114. (CURRENTLY AMENDED) The apparatus of claim 111, further comprising:
2 means for communicating with a the memory;
3 means for storing intermediate operands; and
4 means for identifying an instruction decode stage for coupling the register file to
5 the first and second input registers of the first and second ALUs to provide intermediate
6 operands as source operands, and for coupling a memory bus to the first input register of
7 the of the first ALU to provide source operands from the memory.

1 115. (CURRENTLY AMENDED) The apparatus of claim 114, further comprising:
2 means for providing, by the first input register of the first ALU, source operands
3 from the memory to both the first input to the first ALU and to the first MUX input,
4 thereby permitting the first input to the second ALU to share the source operands from
5 the memory directly from the first input register of the first ALU.

1 116. (PREVIOUSLY PRESENTED) The apparatus of claim 111, further comprising:
2 means for sharing a source operand stored in a first input register of the first ALU
3 at the first and second ALU substantially simultaneously.

1 117. (CANCELLED)

- 1 118. (PREVIOUSLY PRESENTED) A processor, comprising:
 - 2 a first ALU, the first ALU capable of addressing a memory to retrieve a source
 - 3 operand from the memory;
 - 4 a second ALU, the second ALU not capable of addressing the memory; and
 - 5 a circuit to couple a first input of the first ALU to a first input of the second ALU
 - 6 to provide an operand retrieved from the memory by the first ALU as an input to the sec-
 - 7 ond ALU.
- 1 119. (PREVIOUSLY PRESENTED) The processor of claim 118, wherein the circuit
- 2 further comprises:
 - 3 a multiplexer, the multiplexer having an input connected to the first input of the
 - 4 first ALU and an output connected to the first input of the second ALU.
- 1 120. (PREVIOUSLY PRESENTED) A processor, comprising:
 - 2 a first ALU, the first ALU capable of addressing a memory to retrieve an operand
 - 3 a second ALU, the second ALU not capable of addressing the memory;
 - 4 a first circuit capable of providing an operand retrieved by the first ALU from a
 - 5 memory as an input to the second ALU;
 - 6 a second circuit capable of providing a result from either the first ALU or the sec-
 - 7 ond ALU as an input to the second ALU; and
 - 8 an instruction set having a register decode value which is capable of selecting as
 - 9 an input to the second ALU either the operand or the result.
- 1 121. (PREVIOUSLY PRESENTED) The processor of claim 120, wherein the circuit
- 2 further comprises:

3 a multiplexer, the multiplexer having a first input connected to the first input of
4 the first ALU, a second input coupled to the result, and an output connected to the input
5 of the second ALU, the multiplexer selecting whether the second ALU receives the oper-
6 and or the result as an input.

1 122. (PREVIOUSLY PRESENTED) The processor of claim 120, further comprising:
2 a circuit providing the result to both the first ALU and the second ALU.

1 123. (PREVIOUSLY PRESENTED) A method for operating a processor, comprising:
2 addressing a memory by a first ALU to retrieve a source operand from the mem-
3 ory;
4 providing a second ALU, the second ALU not capable of addressing the memory;
5 and
6 coupling a first input of the first ALU to a first input of the second ALU to pro-
7 vide an operand retrieved from the memory by the first ALU as an input to the second
8 ALU.

1 124. (PREVIOUSLY PRESENTED) The method of claim 123, further comprising:
2 connecting an input of a multiplexer to the first input of the first ALU; and
3 connecting an output of the multiplexer to the first input of the second ALU.

1 125. (PREVIOUSLY PRESENTED) A method for operating a processor, comprising:
2 providing an operand retrieved by a first ALU from a memory as an input to a
3 second ALU, the second ALU not capable of addressing the memory;

4 providing a result from either the first ALU or the second ALU as an input to the
5 second ALU; and

6 selecting, by an instruction set having a register decode value, either the operand
7 or the result as an input to the second ALU.

1 126. (PREVIOUSLY PRESENTED) The method of claim 125, further comprising:
2 connecting a first input of a multiplexer to the first input of the first ALU;
3 connecting a second input of the multiplexer to the result; and
4 connecting an output of the multiplexer to the input of the second ALU, the mul-
5 tiplexer selecting whether the second ALU receives the operand or the result as an input.

1 127. (PREVIOUSLY PRESENTED) The method of claim 125, further comprising:
2 providing the result to both the first ALU and the second ALU.